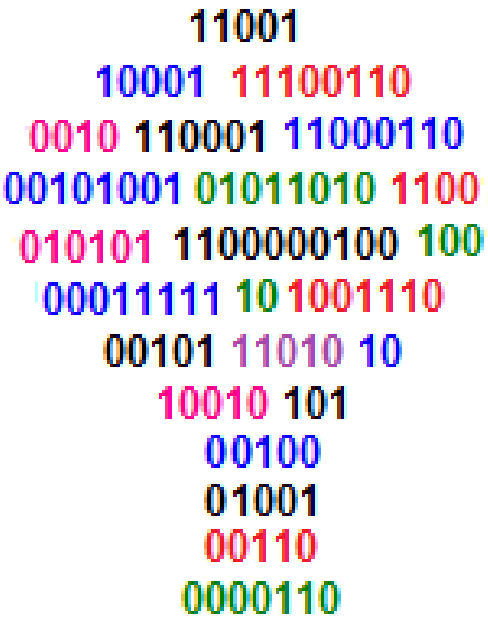
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS



Data Structure

Τεχνική Αναφορά

ΔΕΥΤΕΡΗΣ ΕΞΑΜΗΝΙΑΙΑΣ ΑΣΚΗΣΗΣ

Αλέξανδρος Μπαμπούνης-Τσάτσος | Δομές δεδομένων | 7/1/2021

Περιεχόμενα

ΣΗΜΕΙΩΣΕΙΣ ΚΩΔΙΚΑ

Η κλάση Node

Η κλάση Node χρησιμοποιείται για να υλοποιήσει τα στοιχεία που ο χρήστης θα εισάγει στο σωρό. Αποτελείται από ένα integer πεδίο data, καθώς και τρία integer πεδία τα οποία υποδηλώνουν τη θέση του κάθε node αντικειμένου στο σωρό μεγίστου, ελαχίστου και στον πίνακα allMyNodes.

Μέθοδος CoolHeap(int mx)

Η μέθοδος CoolHeap δέχεται έναν ακέραιο θετικό αριθμό ο οποίος ορίζει το μέγεθος του σωρού. Δημιουργεί δύο πίνακες τύπου Node για τους σωρούς ελαχίστου και μεγίστου αντίστοιχα και έναν πίνακα γενικού σκοπού για εσωτερική χρήση με την ονομασία allMyNodes. Ο τελευταίος περιέχει όλα τα στοιχεία που έχουν εισαχθεί στο σωρό coolHeap και λειτουργεί ως ευρετήριο κόμβων για τις υπόλοιπες εργασίες του.

Μέθοδος isEmpty()

Η μέθοδος ελέγχει και επιστρέφει Boolean τιμή αναλόγως με το αν ο σωρός coolHeap είναι άδειος ή όχι.

Μέθοδος Integer insertMax(int key)

Επιχειρεί εισαγωγή στο σωρό του δεδομένου κλειδιού key. Αν ο σωρός έχει λιγότερα από mx κλειδιά, η εισαγωγή είναι επιτυχής και η μέθοδος επιστρέφει null. Αν ο σωρός αποθηκεύει mx κλειδιά, το ελάχιστο εκ των mx+1 κλειδιών (συμπεριλαμβανομένου του δεδομένου κλειδιού key) απομακρύνεται και επιστρέφεται. Πιο συγκεκριμένα, αρχικά πραγματοποιείται έλεγχος για να διαπιστωθεί αν υπάρχει ελεύθερος χώρος στο σωρό. Αν ναι, τότε παράγεται το αντικείμενο node και εισάγεται στην τελευταία ελεύθερη θέση των σωρών μεγίστου και ελαχίστου και αμέσως καλούνται οι μέθοδοι trickleupForMaxHeap και trickleupForMinHeap αντίστοιχα ώστε να μετακινηθεί το νεοεισερχόμενο στοιχείο στη σωστή θέση και να ισχύει η ιδιότητα του σωρού. Αν ο σωρός δεν έχει ελεύθερη θέση, τότε ελέγχεται αν το key είναι μικρότερο της ρίζας του σωρού ελαχίστου (που είναι το μέχρι τότε ελάχιστο) και αν ναι, το νέο στοιχείο επιστρέφεται και δεν εισάγεται. Αν δεν είναι μικρότερο της ρίζας του σωρού ελαχίστου τότε καλείται η removeMin η οποία διαγράφει και επιστρέφει το ελάχιστο όλων των κλειδιών και ακολούθως εκτελείται η εισαγωγή του νέου κλειδιού. Κάθε φορά που εκτελείται εισαγωγή ή διαγραφή στοιχείου από το σωρό και καλούνται οι trickleup ή trickledown μέθοδοι, ενημερώνονται και τα πεδία των θέσεων του κάθε κόμβου στους αντίστοιχους σωρούς και στον γενικό πίνακα allMyNodes.

Πολυπλοκότητα: Η πολυπλοκότητα της insertMax είναι O(logn). Αυτό προκύπτει ως εξής: Στον πρώτο έλεγχο (γραμμή 38) εκτελείται μια εισαγωγή σε σωρό μεγίστου και μία σε ελαχίστου. Κάθε μια από αυτές έχει πολυπλοκότητα O(logn) άρα στη χειρότερη περίπτωση έχουμε O(logn) για το πρώτο σκέλος. Στο δεύτερο σκέλος του πρώτου ελέγχου (γραμμή 57 και κάτω) εκτελείται η removeMin με O(logn) και πάλι εισαγωγή του στοιχείου και τακτοποίηση των επιμέρους σωρών δηλαδή πάλι O(logn). Άρα στη χειρότερη περίπτωση έχουμε πολυπλοκότητα 3*logn άρα O(logn).

Μέθοδος Integer insertMax(int key)

Επιχειρεί εισαγωγή στο σωρό του δεδομένου κλειδιού key. Αν ο σωρός έχει λιγότερα από mx κλειδιά, η εισαγωγή είναι επιτυχής και η μέθοδος επιστρέφει null. Αν ο σωρός αποθηκεύει mx κλειδιά, το μέγιστο εκ των mx+1 κλειδιών (συμπεριλαμβανομένου του δεδομένου κλειδιού key) απομακρύνεται και επιστρέφεται. Η μέθοδος είναι πανομοιότυπη με την insertMax που περιγράψαμε παραπάνω με τις εξής αλλαγές: Στο δεύτερο σκέλος του πρώτου ελέγχου (γραμμή 106 και κάτω) ελέγχεται αν το key είναι μεγαλύτερο της ρίζας του σωρού μεγίστου (όπου βρίσκεται το μέχρι τότε μέγιστο στοιχείο) και αντί να εκτελείται η removeMin, εκτελείται η removeMax. Όλες οι άλλες διαδικασίες παραμένουν ίδιες με αυτές της insertMax. Συνεπώς και εδώ η πολυπλοκότητα είναι O(logn).

Μέθοδος Integer removeMax()

Η removeMax ελέγχει αν ο σωρός είναι άδειος και επιστρέφει null αλλιώς διαγράφει το μέγιστο στοιχείο του σωρού. Καταλαβαίνουμε ότι πρέπει να διαγράψουμε τη ρίζα του σωρού μεγίστου. Αρχικά αποθηκεύω τον κόμβο ρίζα στη μεταβλητή lastMax ώστε αργότερα να μπορώ να την ξαναχρησιμοποιήσω αν χρειαστεί. Θέτω ως ρίζα το τελευταίο στοιχείο του τελευταίου επιπέδου και αμέσως ενημερώνω το πεδίο του positionInMaxHeap. Ακολούθως καλώ την trickledownForMaxHeap στη ρίζα ώστε να επαναφέρω την κατάσταση σωρού.
Ωστόσο αφού αφαιρέσαμε ένα στοιχείο από το σωρό μεγίστου θα πρέπει το ίδιο να αφαιρεθεί και από το σωρό ελαχίστου. Για αυτό το σκοπό χρησιμοποιώ το πεδίο positionInMinHeap του αντικειμένου lastMax που αναφέρθηκε παραπάνω. Μέσω αυτού μπορώ να βρω τη θέση του παλαιού μεγίστου κόμβου στο σωρό ελαχίστου το οποίο και αφαιρώ. Εν τέλει, αποκαθιστώ την ιδιότητα του σωρού στο σωρό ελαχίστου με την κλήση της trickleupForMinHeap στη θέση του lastMax στο σωρό ελαχίστου.
Πολυπλοκότητα: Η πολυπλοκότητα της removeMax αναλύεται σε δυο διαδικασίες διαγραφής οι οποίες έχουν πολυπλοκότητα O(logn), άρα στη χειρότερη περίπτωση είναι 2*logn, άρα συνολικά O(logn).

Μέθοδος Integer removeMin()

Η removeMin ελέγχει αν ο σωρός είναι άδειος και επιστρέφει null αλλιώς διαγράφει το ελάχιστο στοιχείο του σωρού. Είναι πανομοιότυπη με την removeMax παραπάνω και έχει την ίδια πολυπλοκότητα με αυτήν με τη διαφορά ότι, ενώ οι διαδικασίες διαγραφών παραμένουν ίδιες και εκτελούνται με την ίδια σειρά, αλλάζουν οι σωροί στους οποίους εκτελούνται.

ΣΥΜΠΛΗΡΩΜΑΤΙΚΑ

Στη κλάση CoolHeap δημιούργησα και τις private μεθόδους

- trickleupForMaxHeap (int index)
- trickledownForMaxHeap (int index)
- trickleupForMinHeap (int index)
- trickledownForMinHeap (int index)

οι οποίες είναι υπεύθυνες για την αποκατάσταση της ιδιότητας σωρού στους αντίστοιχους σωρούς μεγίστου και ελαχίστου. Έχουν παρθεί από τις διαφάνειες του μαθήματος και η πολυπλοκότητα τους είναι O(logn) καθώς στη χειρότερη περίπτωση θα χρειαστεί να διασχίσουν το μεγαλύτερο μονοπάτι του δέντρου, άρα O(logn).